SlaBins: Fisheye Depth Estimation using Slanted Bins on Road Environments

Jongsung Lee[1], Gyeongsu Cho[1]*, Jeongin Park[1]*, Kyongjun Kim[1]*, Seongoh Lee[1]*,
Jung-Hee Kim[2], Seong-Gyun Jeong[2], Kyungdon Joo[1]†
[1]Artificial Intelligence Graduate School, UNIST, [2]42dot Inc.
{syniez, threedv, jeonginpark, kimkj38, solee, kyungdon}@unist.ac.kr
{junghee.kim, seonggyun.jeong}@42dot.ai

Abstract

Although 3D perception for autonomous vehicles has focused on frontal-view information, more than half of fatal accidents occur due to side impacts in practice (e.g., *T-bone crash). Motivated by this fact, we investigate the problem of side-view depth estimation, especially for monocular fisheye cameras, which provide wide FoV information. However, since fisheye cameras head road areas, it observes road areas mostly and results in severe distortion on object areas, such as vehicles or pedestrians. To alleviate these issues, we propose a new fisheye depth estimation network,* `SlaBins`*, that infers an accurate and dense depth map based on a geometric property of road environments; most objects are standing* (i.e., *orthogonal) on the road environments. Concretely, we introduce a slanted multi-cylindrical image (MCI) representation, which allows us to describe a distance as a radius to a cylindrical layer orthogonal to the ground regardless of the camera viewing direction. Based on the slanted MCI, we estimate a set of adaptive bins and a per-pixel probability map for depth estimation. Then by combining it with the estimated slanted angle of viewing direction, we directly infer a dense and accurate depth map for fisheye cameras. Experiments demonstrate that* `SlaBins` *outperforms the state-of-the-art methods in both qualitative and quantitative evaluation on the SynWoodScape and KITTI-360 depth datasets. For more information, you can visit our project page* https://syniez.github.io/SlaBins/.

1. Introduction

With the advent of autonomous driving, the importance of 3D perception keeps growing for safety [20, 25, 44]. For example, ADAS systems pervading our daily driving already, such as lane-keeping aid and forward collision-avoidance assist, depend on various sensor modalities to observe 3D information [43, 5]. Here, one interesting point is that various sensor modalities for 3D perception, including many datasets related to autonomous driving, are focusing on the frontal view predominantly [43, 38, 2, 6]. We can easily deduce the reason by our common sense that it is important to keep an eye on the front while driving. However, according to the National Transportation Highway Safety Administration [1], 51% of fatal car accidents correspond to T-bone crashes in which one car hits the side of another, which is the second highest type of accident. This statistic supports that side-view 3D perception is important to assist drivers because they are hard to pay attention to the side view due to the physical characteristics of the people looking ahead and their position characteristics.

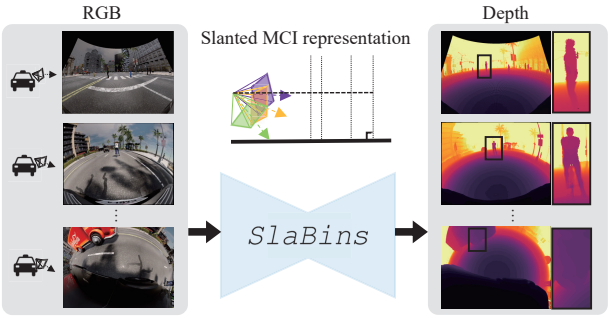


Figure 1: **`SlaBins` on the SynWoodScape dataset [34].** `SlaBins` predicts dense and accurate depth maps regardless of the viewing directions using the slanted MCI representation.

Motivated by this fact, 3D perception for side-view, especially fisheye-based depth estimation, has started to gain the spotlight [31, 16, 41]. Fisheye cameras have several advantages for 3D perception in road scenes. Specifically, fisheye cameras provide broader FoV information (typically greater than 180°) in a single image, which allows us to observe road information (*e.g.,* road, vehicles, and pedestrians) at a low cost. In addition, we can directly utilize perspective camera 3D perception approaches because mathematical mutual conversion between two camera models is possible. Thus, if we can obtain side-view depth informa-

*Equal contribution (alphabet order).
†Corresponding author.

tion using a single fisheye camera, we can efficiently handle various vision tasks for autonomous driving.

However, despite the advantages of fisheye cameras, monocular fisheye camera-based side-view depth estimation is a challenging problem due to significant distortion. In addition to inherent issues of monocular depth estimation, such as scale ambiguity [8], severe distortion of fisheye cameras makes depth estimation more difficult by distorting scene objects (*e.g.,* close objects look closer and distant objects look more distant abnormally). Particularly, this distortion problem could be more severe in road environments because fisheye cameras for ADAS systems generally look to the road areas, and vital objects such as vehicles and pedestrians are located at the edge of the image. That is, most regions of the image correspond to the road areas and objects contain very few pixels, so depth estimation becomes more challenging.

To alleviate this problem, we pay attention to the geometric property of road environments; most objects are standing on the road environments. Based on this geometric characteristic, we can easily imagine the visible region of each object as a planar segment orthogonal to the road region. This approximation is connected to a multi-plane image (MPI) concept [36] that represents a given scene as the composition of multi-layered images according to depth. Note that one significant difference is that a fisheye camera looks down to the road region in our case (*i.e.,* camera viewing direction is not parallel to the ground), unlike a camera points to a given scene in the general MPI concept. Thus, a naïve MPI concept is not directly applicable.

In this work, we propose a new fisheye depth estimation framework, `SlaBins`, that exploits the geometric property of the road environments (see Fig. 1). To be specific, we adopt a multi-cylindrical image (MCI) representation[1]; especially, we introduce a slanted MCI of which the cylindrical layer is defined as orthogonal to the road ground regardless of camera viewing direction. Based on the slanted MCI, we estimate adaptive bins and the per-pixel probability of the slanted cylindrical layers, which allows us to implicitly preserve the geometric property of the road environments and improve the depth quality near the boundary between objects and the ground region. In addition, we independently estimate the slanted angle of the camera viewing direction *w.r.t.* the road ground. Then, by combining this angle with the estimated depth information in the slanted MCI domain, we can directly compute a dense and accurate depth map for fisheye cameras. We validate the proposed `SlaBins` framework on the SynWoodScape dataset [34] and KITTI-360 depth dataset (a modified dataset of KITTI-360 [24] for our scenario), where `SlaBins` outperforms the baseline methods [3, 20, 21] and shows potential as scene representations for downstream tasks, *e.g.,* segmentation.

In a nutshell, our contributions are as follows:

- We propose a new fisheye depth estimation framework, `SlaBins`, which infers dense depth based on the geometric property of road environments.
- We newly introduce a slanted MCI representation as an intermediate road scene representation, which allows us to describe a given scene regardless of viewing direction and improve the depth quality on the boundary areas between objects and the road.
- By decoupling the slanted angle of an input image, we force the proposed model to learn the slanted MCI depth information and then seamlessly compute a fisheye depth map in a slanted-aware regression way.

2. Related Work

Monocular depth estimation. With the advent of breakthroughs in deep neural networks, such as CNN and transformer architecture, monocular depth estimation has shown impressive performance [15, 33, 21, 32, 23, 18, 40, 11, 9, 12, 29, 19, 16, 17, 20]. As one of the recent trends, several methods formulate a depth estimation task as a combination of classification and regression problems, which enables training the network more easily and shows stable performance [14, 3, 4]. For this formulation, they commonly use an MPI according to depth, which is popularly used in novel-view synthesis tasks [22, 39, 13]. The MPI notation represents depth as quantized depth values *w.r.t.* the uniformly quantized planes and residuals from mapped planes.

To improve the uniformly quantized planes by uniform bins, recent research works [3, 4] present a network that predicts adaptive bins depending on the input scene and local regions in the image. AdaBins [3] computes adaptive bin-centers and per-pixel probabilities relying on each input image. Inspired by AdaBins, LocalBins [4] improves depth quality by considering both the global histogram and local-region histograms of each input image. Nevertheless, the predicted depth from the above models also depends on the camera's pose because they predict depth parallel with the camera's principal axis. This constraint makes it hard to adopt the previous models into fisheye depth estimation, which has a large FoV and uses the Euclidean distance instead of depth. In this work, we introduce and exploit a slanted MCI representation that is invariant to the viewing angle of fisheye cameras *w.r.t.* road environments.

Fisheye depth estimation. Despite increasing demand for fisheye cameras in the industry, like ADAS systems, there are only a few previous research works in fisheye depth estimation fields [20, 17, 19, 16, 40]. Kumar *et al.* [19] propose

[1]Basically, depth in the fisheye model is defined as the Euclidean distance, which is connected to a multi-spherical image (MSI). In this work, We adopt an MCI representation that combines the geometrical suitability of MPI and the fisheye model fitness of MSI.

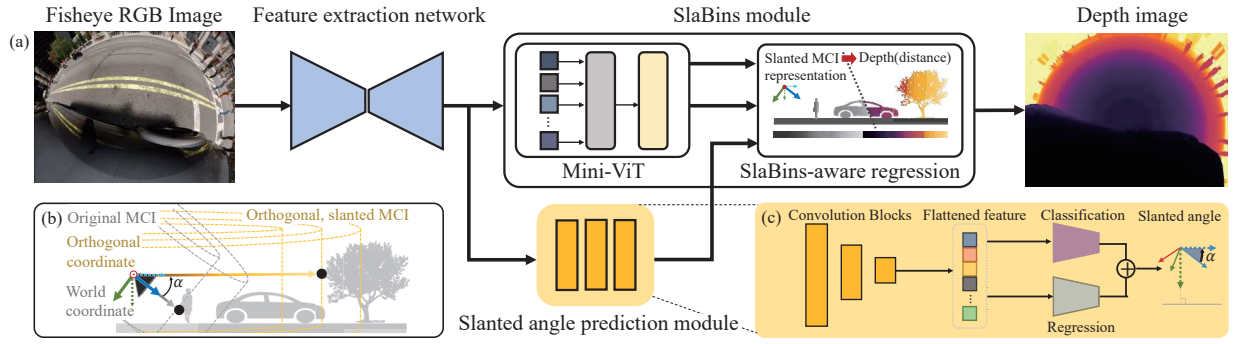


Figure 2: **Overview of the proposed `SlaBins`**. (a) Given an input fisheye RGB image, `SlaBins` framework first extracts features by the encoder-decoder block. The extracted features are used as input to the slanted angle prediction module and the SlaBins module to get a depth image. (b) Illustration of the coordinates and MCI representations defined in our model. (c) Illustration of the slanted angle predictor.

a camera-aware depth estimation network to handle the severe distortion of fisheye cameras. They encode the camera's intrinsic parameters as a tensor named camera geometry tensor and use it in the training step for camera-aware depth estimation in fisheye images. With camera geometry tensor, they propose a multi-task network for fisheye cameras, called OmniDet [20], that simultaneously estimates depth estimation, motion, semantic segmentation, and object detection for synergy. However, since OmniDet relies on the partial region in the whole image, they may miss information on image edges. Using a whole image, Yan *et al.* [40] distill depth ordering information from a teacher network trained on the large-scale dataset to train in fisheye images, but FoV they cover is not large (*e.g.,* vFoV=66°). In contrast to previous approaches, we propose a fisheye depth estimation network that predicts fisheye image depth without any information loss such as crop, and narrow FoV.

Geometric cues for depth estimation. One way to improve fisheye depth estimation is to exploit geometric cues between the camera and the world. Liu *et al.* [26] use a ground plane to get depth priors using camera elevation and camera parameters. Moreover, Mahjourian *et al.* [29] predict camera ego-motion and compute loss on both the 3D point cloud domain and the 2D image domain. In addition to driving scenes, several works use plane coefficients to give consistency on plane regions in depth estimation [30, 27]. Patil *et al.* [30] estimate plane coefficients instead of depth and then combine plane coefficients into depth to use plane constraints (*e.g.,* co-planarity, plane normals) in 3D space. Inspired by the previous works [3, 30], we design a network that estimates a dense and accurate depth map for fisheye cameras based on the slanted MCI representation using some geometric constraints (*i.e.,* orthogonality between ground and object).

3. Proposed Framework

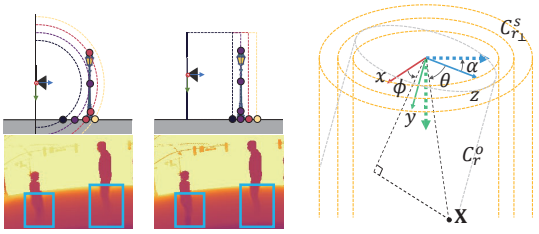
Inspired by the geometric property of the road environments, we propose a new fisheye depth estimation framework (see Fig. 2 for the overview). Specifically, we introduce a slanted MCI representation, which allows us to describe a multi-layer cylindrical image orthogonal to the road ground regardless of the viewing direction of the camera. Based on the slanted MCI representation, our SlaBins module estimates the adaptive bin widths and per-pixel probability map in the orthogonal coordinate, which provides depth information invariant to the camera viewing direction. We then combine the estimated depth information with the slanted angle estimated by the slanted angle prediction module. Through this process, we can directly estimate a dense depth map for fisheye cameras. Our approach is built upon the previous AdaBins framework [3]. So, we call our method `SlaBins`.

3.1. Slanted MCI Representation

As intermediate scene representation, we use multi-layer images for fisheye depth estimation. Here, we briefly compare two multi-layer image representations for fisheye images: multi-spherical image (MSI) and multi-cylindrical image (MCI). We then present a slanted MCI representation, which is a suitable representation of our problem.

MSI *vs*. MCI. By nature of fisheye cameras covering over 180°, it is common to use the Euclidean distance as its depth; that is, a layer with the same depth value forms a sphere in the fisheye domain. Thus, we can represent a multi-layer image for fisheye cameras as an MSI in the spherical coordinate, which is a naïve representation. However, this naïve MSI representation has inherent limitations for fisheye depth estimation. We observed that such a representation can result in ambiguous depth boundaries between the ground and object regions, as shown in Fig. 3a. It may neutralize the benefit of multi-layer images by separating an object into two different layers, for example.

In the MCI representation, a cylindrical layer encodes the same Euclidean distance *w.r.t.* the y-axis. In other words, we can represent an object, such as a pedestrian on the road, in a single cylindrical layer (see Fig. 3b). Further-



(a) (b) (c)

Figure 3: **MSI vs. MCI representation**. Illustration of an example scene and zoomed results with (a) MSI and (b) MCI, where we can observe more accurate depth in the slanted MCI representation near the boundary between pedestrians and the ground. (c) Relationship between the original cylindrical layer C^o_r (gray-colored cylinder) in the world coordinate (arrows) and the slanted cylindrical layer $C^s_{r_\perp}$ (orange-colored cylinders) in the orthogonal coordinate (dashed arrows).

more, we can directly exploit the geometric properties of the road environment: most objects are standing (*i.e.,* orthogonal) on the road environment. For this reason, we adopt the MCI as our intermediate scene representation.

Slanted MCI representation. Let $\mathbf{X}$ be a 3D point in the world coordinate. Then, we can represent $\mathbf{X}$ according to the coordinate systems:

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \rho \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{bmatrix} = \rho \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix}, \tag{1}$$

where $\hat{\mathbf{X}} = [\hat{X}, \hat{Y}, \hat{Z}]^\top$ is the normalized unit vector (*i.e.,* ray direction), ρ indicates the Euclidean distance (*i.e.,* depth in the fisheye domain), and θ and ϕ denote the elevation and azimuth angles, respectively.

Based on the cylindrical representation, we can form a cylindrical layer according to a radius r as:

$$C^o_r = \{\mathbf{X} \mid X^2 + Z^2 = r^2; \mathbf{X} \in \mathbb{R}^3\}. \tag{2}$$

We call C^o_r the original cylindrical layer at r. This original MCI is inherently suitable for road environments and shows less distortion than MSI on ambiguous depth boundaries. However, depending on the principal axis of cameras (*i.e.,* viewing direction) *w.r.t.* the road environments, the original MCI may hinder layered scene representation like the MSI.

To alleviate this issue, we consider the 1D angle α between the principal axis and the road environments. Using α, we can implicitly describe the slanted MCI representation that is orthogonal to the road areas (see Fig. 3c). To this end, let $\mathbf{X}_\perp$ be a 3D point in the orthogonal coordinate, of which the XZ plane is parallel to the ground. Similarly to Eq. (2), we can form a cylindrical layer according to a radius $r_\perp$ in the orthogonal coordinate as:

$$C^s_{r_\perp} = \{\mathbf{X}_\perp \mid X^2_\perp + Z^2_\perp = r^2_\perp; \mathbf{X}_\perp \in \mathbb{R}^3\}, \tag{3}$$

which corresponds to a slanted cylindrical layer in the world (camera) coordinate (see Fig. 2b for coordinate systems).

By α, we can derive the relationship between $\mathbf{X}_\perp$ in the orthogonal coordinate and $\mathbf{X}$ in the world coordinate by:

$$\mathbf{X}_\perp = \begin{bmatrix} X_\perp \\ Y_\perp \\ Z_\perp \end{bmatrix} = \begin{bmatrix} X \\ Y\cos\alpha + Z\sin\alpha \\ -Y\sin\alpha + Z\cos\alpha \end{bmatrix}. \tag{4}$$

Then, we can deduce the following relationship between a 3D point $\mathbf{X}_\perp$ on $C^s_{r_\perp}$ and its spherical representation in the camera coordinate by using Eqs. (3) and (4):

$$\begin{aligned} r^2_\perp = X^2_\perp + Z^2_\perp &= X^2 + (-Y\sin\alpha + Z\cos\alpha)^2, \\ &= \rho^2\left(\hat{X}^2 + (-\hat{Y}\sin\alpha + \hat{Z}\cos\alpha)^2\right). \end{aligned} \tag{5}$$

Furthermore, we can convert the relationship in Eq. (5) to a function f as:

$$f(\hat{\mathbf{X}}, r_\perp, \alpha) = \rho, \tag{6}$$

where $r_\perp$ denotes a radius in the orthogonal coordinate. The function f represents that if we know slanted angle α and a radius $r_\perp$ at a certain pixel location on the fisheye domain, we can directly compute the corresponding depth value ρ. A more detailed explanation for Eq. (6) is available in the supplementary material. Using this relation, we implicitly model the slanted MCI representation inside the network as scene representation.

3.2. SlaBins framework

Based on the slanted MCI representation (*cf.* Sec. 3.1), the proposed `SlaBins` framework estimates an accurate and dense depth map. `SlaBins` consists of three modules: Encoder-decoder block, slanted angle predictor, and SlaBins module, as shown in Fig. 2.

Concretely, given the input fisheye image, the encoder-decoder block extracts a feature map containing useful information for depth prediction. Using the extracted feature map, the SlaBins module estimates several key information, such as adaptive bin-centers and per-pixel probability map in the slanted MCI representation (*i.e.,* the orthogonal coordinate), using a transformer architecture. It should be worth noticing that by virtue of this SlaBins module, we can implicitly preserve the geometric relationship of objects on the road environments. Then, by combining with the slanted angle estimated by the slanted angle prediction module, we can directly regress a fisheye depth map.

Encoder-decoder block. Given input fisheye image $x_i \in \mathbb{R}^{H\times W\times 3}$, where H and W denote the height and the width of the input, our encoder-decoder block first extract features. We use the pre-trained EfficientNet B5 [37] as the backbone for the encoder and standard feature upsampling layers for the decoder, which returns decoded features $x_d \in \mathbb{R}^{h\times w\times C_d}$, where $h = H/2$, $w = W/2$, and C_d indicates the feature dimension.

Slanted angle prediction module. We design compact MLP-based classification and regression modules that predict α from x_d in a two-step. The classifier first classifies

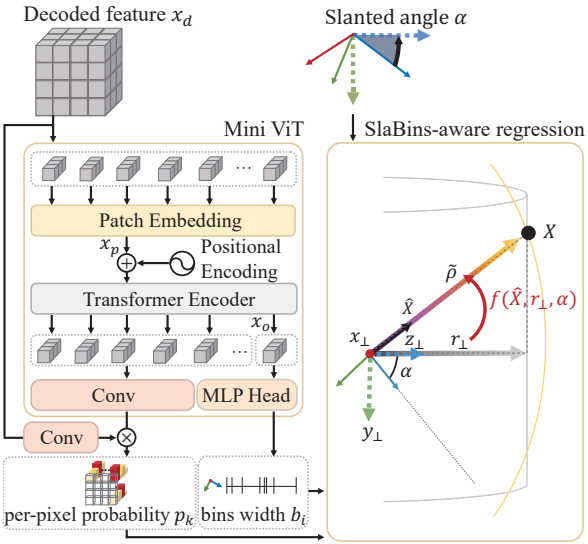


Figure 4: **The details of the SlaBins module**. Our SlaBins module consists of two components: One is the mini-ViT block that extracts the range-attention map and bin widths, which are invariant to the slanted angle α from the decoded feature. The other is the SlaBins-aware regression. This module receives the outputs of mini-ViT and the slanted angle α as inputs and directly converts the orthogonally estimated depth $r_\perp$ for an arbitrary 3D point $\mathbf{X}$ into the depth ρ of the spherical coordinate.

α in units of 5°, and the following regressor estimates the offset in the range of $(-2.5°, 2.5°)$. This two-step angle prediction enables us to estimate an accurate viewing direction angle and train more efficiently (see Fig. 2c).

Specifically, x_d is downsampled to $1/4$ size with two 3×3 convolution layers and average pooling. Downsampled features are flattened (we denote the flattened feature as x_f) and x_f is used as input of classifier and regressor, which are composed of 3 FC layers followed by softmax and 4 FC layers, respectively. The output of the classifier is a 19-dimensional vector and that of the regressor is a normalized angle offset in the range of $(-1°, 1°)$.

SlaBins module. Given decoded features x_d, the proposed SlaBins module first learns intermediate depth information, such as bin widths and probability maps in the slanted MCI representation, which are invariant to the viewing direction (*i.e.,* slanted angle α), as shown in Fig. 4. To this end, we adopt mini-ViT (mViT) architecture in [3].

The decoded features x_d are fed into the mViT with learnable positional encodings. To make a fixed-size patch-embedded input, we embed the decoded features to the C_p dimensional flattened tensor $x_p \in \mathbb{R}^{S\times C_p}$ using patch embedding before feeding into the mViT (here, S indicates $w\times h/p^2$ and p denotes the size of patch and stride). Then, the sum of flattened tensor x_p and positional encodings are fed into the mViT and the output of the mViT has the same shape as input tensor $x_o \in \mathbb{R}^{S\times C_p}$. The first MLP head of the output x_o means slanted bin widths and it is normalized into b_i for further depth regression. The other features followed by the 1×1 convolution layer perform pixel-wise dot product with a 3×3 convolution output of x_d to get the range-attention map. Softmax function with 1×1 convolution layer is used to represent the range-attention map as a probability of each pixel p_k in N bin-centers. These normalized slanted bin-width b_i and per-pixel probability p_k with α are used for SlaBins-aware regression.

Our final depth $\tilde{\rho}$ is computed by the linear combination of the two outputs from mViT (*i.e.,* slanted bin-width b_i and per-pixel probability p_k) and the transformation function in Eq. (6). To compute final depth, slanted depth bin-centers c^s are computed with normalized bin-width b_i (*i.e.,* $\sum b_i{=}1$):

$$c^s(b_i) = r_\perp^{min} + (r_\perp^{max} - r_\perp^{min})(b_i/2 + \Sigma_{j=1}^{i-1} b_j), \quad (7)$$

where $r_\perp^{min}$ and $r_\perp^{max}$ mean the lower and upper bound of the radius in slanted MCI representation. Then, orthogonal coordinate radius $r_\perp$ is calculated using the cumulative sum of c_k^s and p_k^s (*i.e.,* $\Sigma_{k=1}^N c^s(b_k)p_k$) and transformed to our final depth $\tilde{\rho}$ using Eq. (6):

$$\tilde{\rho} = f\left(\hat{\mathbf{X}}, \Sigma_{k=1}^N c^s(b_k)p_k, \alpha\right), \quad (8)$$

where $\hat{\mathbf{X}}$ is a known normalized vector computed by pixel location and camera intrinsic, and α denotes the estimated slanted angle by the slanted angle predictor.

3.3. Loss function

Here, we introduce a set of loss functions used for training `SlaBins`. Specifically, we compute loss in both orthogonal and spherical coordinates to make our `SlaBins` framework can estimate fisheye depth that has better discrimination between objects and road environments.

Bin loss. Following [3], we use Chamfer loss[7] to make adaptive bins:

$$\mathcal{L}_{bins} = chamfer(r_\perp, c^s(\tilde{\mathbf{b}})) + chamfer(c^s(\tilde{\mathbf{b}})), r_\perp), \quad (9)$$

where $\tilde{\mathbf{b}}$ indicates the predicted slanted bin widths from the SlaBins module, and $r_\perp$ is the ground truth cylinder radius in the slanted MCI representation. In our case, to generate slanted bins (*i.e.,* bins in the orthogonal coordinate), the ground-truth depth is converted to the ground truth radius $r_\perp$ at the orthogonal coordinate by Eq. (5) and the slanted MCI.

Pixel-wise depth loss. We apply a scaled version of the Scale-Invariant (SI) log loss introduced by Eigen *et al.* [42]:

$$\mathcal{L}_{\text{pixel}} = \beta\sqrt{\frac{1}{T}\Sigma_i g_i^2 - \frac{\lambda}{T^2}\left(\Sigma_i g_i\right)^2} \quad (10)$$

where $g_i = \log\tilde{\rho} - \log\rho$ with the ground truth depth ρ and predicted depth $\tilde{\rho}$. T denotes the number of pixels containing valid ground truth values. Following [3], we set $\lambda = 0.85$ and $\beta = 10$ for all our experiments.

Table 1: **Quantitative evaluation on SynWoodScape** [34].

Method	RMSE ↓	RMSE log ↓	Abs Rel ↓	Sq Rel ↓	$\delta 1 \uparrow$	$\delta 2 \uparrow$	$\delta 3 \uparrow$
OmniDet* [20]	5.623	0.740	1.080	3.133	0.031	0.064	0.753
BTS [21]	2.998	0.325	0.179	0.459	0.653	0.845	0.932
AdaBins [3]	1.524	0.104	0.070	0.127	0.971	0.993	0.997
SlaBins (ours)	**1.040**	**0.056**	**0.022**	**0.055**	**0.988**	**0.997**	**0.999**

Table 2: **Quantitative evaluation on KITTI-360 depth**.

Method	RMSE ↓	RMSE log ↓	Abs Rel ↓	Sq Rel ↓	$\delta 1 \uparrow$	$\delta 2 \uparrow$	$\delta 3 \uparrow$
OmniDet* [20]	1.917	0.150	0.086	0.246	0.908	0.975	0.992
BTS [21]	1.787	0.153	0.093	0.243	0.906	0.976	0.992
AdaBins [3]	1.590	**0.129**	**0.074**	0.185	0.940	**0.984**	**0.994**
SlaBins (ours)	**1.537**	0.130	0.077	**0.177**	**0.941**	0.983	**0.994**

Slanted angle loss. As mentioned in Sec. 3.2, we predict slanted angle α through a two-step module: classification and regression. For the classification module, we use the cross-entropy loss $\mathcal{L}_{CE}$. As for the regression module, we utilize the mean squared error (MSE) loss $\mathcal{L}_{MSE}$ between the estimated residual and the ground truth residual. We define the sum of the CE loss and MES loss as the slanted angle loss:

$$\mathcal{L}_{angle} = \mathcal{L}_{CE} + \mathcal{L}_{MSE}. \tag{11}$$

Total loss. Finally, the total loss for our problem is:

$$\mathcal{L}_{total} = \gamma \mathcal{L}_{bins} + \mathcal{L}_{pixel} + \mathcal{L}_{angle}, \tag{12}$$

where γ is a balancing parameter (we set $\gamma = 0.1$).

4. Experiments

We evaluate the proposed method from various perspectives. We introduce datasets and evaluation metrics in Sec. 4.1 and implementation details in Sec. 4.2. We then offer quantitative and qualitative comparisons to state-of-the-art methods in Sec. 4.3 and an ablation study in Sec. 4.4.

4.1. Datasets and evaluation metrics

For evaluation, we use two fisheye depth datasets: the SynWoodScape dataset [34], which is a synthetic dataset, and the KITTI-360 depth dataset, which is a modified dataset of real-world KITTI-360 [24] for our scenario. Note that we additionally augment two datasets for our purposes and will release these datasets for research communities.

SynWoodScape dataset [34]. The SynWoodScape dataset is a synthetic surround-view fisheye camera dataset for autonomous driving, which is a synthetic version of the real-world WoodScape dataset [42][2]. This dataset provides pairs of synthetic fisheye images and depth maps for the front, left, right, and rear of the driving situation. Each view contains 500 images with 190° FoV and a size of 1280×966.[3]

Additionally, we augment the SynWoodScape dataset in terms of the viewing direction of fisheye cameras to validate the proposed framework. Specifically, we warp fisheye images and the corresponding depth maps by changing a total of nine viewing directions, of which rotation angles *w.r.t.* the x-axis are between 0° and 90° (between the viewing direction and the normal of the ground), including the original viewing direction. So, we have 18K fisheye images in total. We split the augmented dataset into 14.4K training, 1.8K validation, and 1.8K test samples. Following the previous work [16], we train the model and evaluate the range of depth up to 40m because fisheye cameras undergo high data compression and can perform up to this range, unlike the KITTI dataset [10] covering up to 80m.

KITTI-360 depth dataset. The KITTI-360 dataset [24] is a suburban driving dataset for 2D/3D semantic and instance segmentation tasks that include various input modalities, semantic instance annotations, and precise localization. KITTI-360 provides fisheye images, raw LiDAR point clouds, and semantic and instance labels with 185° FoV of left and right in the driving situation, with a resolution of 1400×1400. Note that since KITTI-360 does not provide the ground truth for fisheye depth estimation, we generate the corresponding ground truth using raw LiDAR data. We call this modified dataset for fisheye depth estimation as *KITTI-360 depth* dataset.

Concretely, we first make dense LiDAR by aggregating LiDAR data of twenty neighbor frames. We then project the dense LiDAR point clouds on the fisheye camera image domain and conduct occlusion handling [18] because of the displacement between the fisheye camera and the LiDAR (see Fig. 6). We then consider depth values, of which distance is less than 40m, as the ground truth. We sample 4K samples of the training set from sequence 0 to 7, 0.4K samples of the validation set from sequence 9, and 0.4K samples of the test set from sequence 10. To make dense depth maps, we resize samples to 700×700. In addition, as we did for SynWoodScape, we augment the viewing directions of the KITTI-360 depth dataset; the angle between the viewing direction and the normal on the ground to 30°, 50°, 70°, and 90° (the original viewing angle). So, we have 19.2K fisheye images in total. We split this augmented dataset into 16K training, 1.6K validation, and 1.6K test samples.

Evaluation metrics. We use the standard depth estimation metrics used in the previous work [21]. For error metrics, we calculate mean absolute relative error (Abs Rel), root mean square error (RMSE), root mean square log error (RMSE log), the root mean square relative error (Sq Rel). For accuracy metrics, we calculate threshold accuracy $\delta_i = 1.25^i$ for $i \in \{1, 2, 3\}$. Generally, the smaller the error, the better, and the higher the accuracy, the better.

4.2. Implementation details

Our proposed method is implemented in PyTorch and trained on 4 RTX A6000 GPUs. Our model is trained for

[2]Unfortunately, the real-world WoodScape dataset [42] does not provide the ground truth depth data.

[3]Currently, the SynWoodScape dataset is partially released to the public. We perform the experiments using the partial dataset.

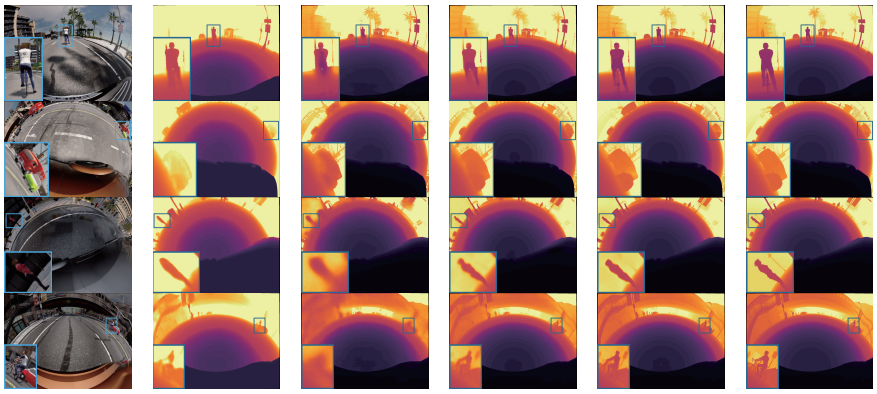


Figure 5: **Qualitative comparison on the SynWoodScape dataset**. Sky-blue boxes are enlarged views of specific objects to show detail. Overall, our method shows distinct boundaries compared to the other models. In particular, we can observe the detailed structure from our results, such as bicycle wheels in the first row and the boundary of two overlapped vehicles in the second row.

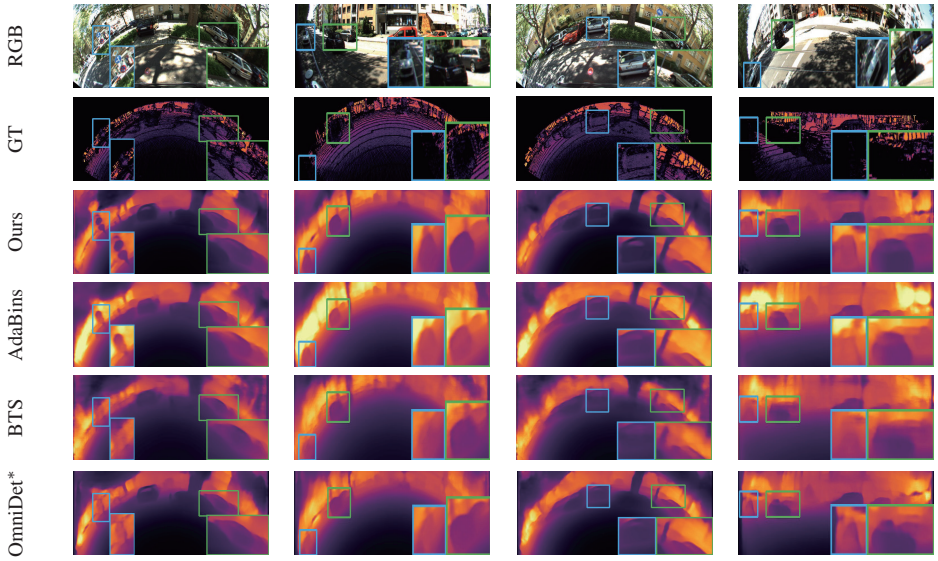


Figure 6: **Qualitative comparison on the KITTI-360 depth dataset**. For visualization, we crop the object region in the fisheye image. In contrast to the other methods, our method shows clear boundaries near the object regions, as shown in sky-blue and green boxes.

10 epochs on SynWoodScape and 20 epochs on KITTI-360 depth using AdamW [28] optimizer with weight-decay 10^{-2}, with a batch size of 8. We use the same 1-cycle policy [35] as AdaBins [3] for the learning rate with max lr = 3.5 $\times 10^{-4}$. For the first 30% of iterations, the linear learning rate warm-up technique is utilized. For the learning rate decay, the cosine annealing learning rate strategy is applied.

4.3. Comparison

We compare our `SlaBins` with several baseline approaches: OmniDet [20], BTS [21], and AdaBins [3]. OmniDet is a fisheye depth estimation method using self-supervised learning. Based on the released code of the authors, we modify OmniDet to a supervised one for fairness (we denote this modified one OmniDet*). As baseline methods for pinhole cameras, we compare BTS and AdaBins, especially AdaBins is a bins-based fundamental method for depth estimation. It should be noted that we target depth estimation fisheye cameras, but we additionally compare it with conventional pinhole camera-based methods due to the lack of fisheye camera-based methods.

Quantitative evaluation. Table 1 shows the quantitative comparisons on the SynWoodScape dataset. The arrow next to the metric indicates the direction of better performance. Overall, `SlaBins` outperforms the other baseline approaches [3, 20, 21]. In particular, RMSE and Abs-Rel errors are significantly improved; 31.7% and 68.6% decreased compared to AdaBins, which shows the effect of the slanted MCI representation in the fisheye domain.

On the KITTI-360 depth dataset, our method still outper-

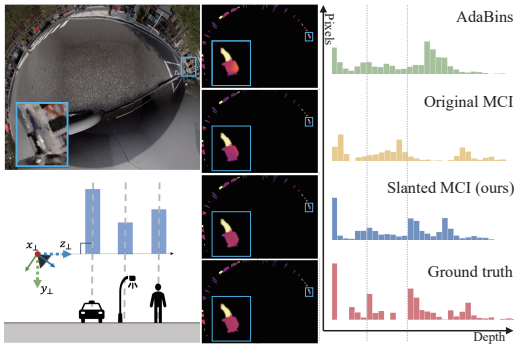


Figure 7: **Depth histogram on the object regions**. *Left*: RGB data (top) and the concept of depth histogram in the orthogonal coordinate (bottom). *Middle*: depth maps of only object regions for each model and the ground truth. *Right*: depth histograms (AdaBins, original MCI, slanted MCI, ground truth in that order).

forms the comparison methods (see Table 2). Specifically, RMSE and Sq-Rel errors are improved; 13.9% and 27.1% decreased compared to BTS and 3.3% and 4.3% decreased compared to AdaBins. Note that even though the KITTI-360 depth dataset captured in suburban scenes does not perfectly fit our scenario of road environments, `SlaBins` still shows competitive performance, which demonstrates the effect of the proposed framework.

Qualitative evaluation. Figure 5 shows the estimated depth maps for the test scenes on SynWoodScape. Due to the nature of fisheye cameras, we can observe ambiguous boundary regions in the comparison methods. Even AdaBins, which is our baseline and a naïve MSI representation by the Euclidean distance of fisheye, shows bleeding depth estimation results near the boundaries between objects and ground regions. In contrast, thanks to the slanted MCI representation, `SlaBins` shows accurate depth results with distinct boundaries regardless of camera viewing directions. Specifically, we can observe that sharp objects with details (*e.g.,* human legs and bike) are preserved in our method, unlike the other methods that smooth object details. In addition, we can observe a similar trend on the KITTI-360 depth dataset, as shown in Fig. 6. Compared to the baseline methods, `SlaBins` shows better depth consistency even on sharp objects (*e.g.,* third and fourth columns in Fig. 6). Additional results are available in the supplementary material.

4.4. Ablation study

Multi-layered representation. We compare the proposed slanted MCI with the MSI used in AdaBins and the original MCI on the orthogonal coordinate. We measure the depth histogram for only object regions on the orthogonal coordinate. As shown in Fig. 7, compared to the other representations, `SlaBins` reveals the most similar trend (*i.e.,* peak points) with the ground truth histogram. Furthermore, we compare slanted MCI and original MCI in terms of depth consistency. Figure 8 shows that the slanted MCI provides

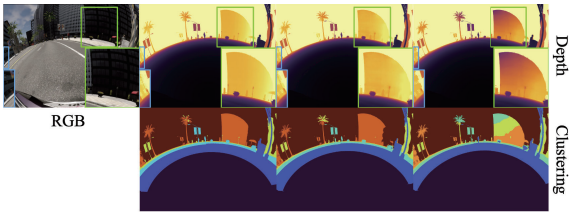


Figure 8: **Slanted MCI *vs.* original MCI in depth and clustering**. All the results including the ground truth are converted to the XZ distance in the orthogonal coordinate.

Table 3: **Ablation study for the slanted angle predictor on SynWoodScape and KITTI-360 depth**. We report the depth error (Abs Rel) and mean angle error. `C` and `R` denote classification-only and regression-only, respectively.

	SynWoodScape		KITTI-360 depth	
Method	Abs Rel ↓	Angle error ↓	Abs Rel ↓	Angle error ↓
`R`	0.075	27.031	0.604	48.32
`C`	0.026	0.842	0.103	5
`C+R` (ours)	**0.022**	**0.031**	**0.077**	**0.017**

more consistent results in object regions both on depth and depth-based clustering results. Hence, we believe that the slanted MCI can provide robust geometric constraints for various tasks in fisheye domains.

Slanted angle prediction. As shown in Table 3, we perform an ablation study for the angle prediction module. Unlike a conventional regression module having the advantage over classifications, in our case, it is hard to converge because of the trade-off with the depth estimation part in the SlaBins module. That is why regression-only module is poor on the angle prediction. On the contrary, our two-step prediction module, composed of classification and residual regression, shows the best performance with stable training.

5. Conclusion

We have proposed a novel fisheye depth estimation network, called `SlaBins`, based on the geometric property of the road environments. We newly introduce the slanted MCI representation, which allows us to represent a given road scene as a set of cylindrical layers orthogonal to the road ground. By virtue of the slanted MCI, our network can implicitly encode scene depth information invariant to the camera viewing direction while preserving boundary information of scene objects *w.r.t.* the road region. Thus, we achieve state-of-the-art performance on monocular fisheye depth estimation.

Acknowledgments. This work was supported by 42dot Inc., Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01336, Artificial Intelligence Graduate School Program (UNIST)) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2021R1C1C1005723).

References

[1] T-Bone Accident Statistics And Other Legal Information. https://www.injurytriallawyer.com/faqs/how-dangerous-are-t-bone-accidents-.cfm. 1

[2] Ashutosh Agarwal and Chetan Arora. Attention Attention Everywhere: Monocular Depth Prediction with Skip Attention. *arXiv preprint arXiv:2210.09071*, 2022. 1

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, 2021. 2, 3, 5, 6, 7

[4] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. LocalBins: Improving Depth Estimation by Learning Local Distributions. In *ECCV*, 2022. 2

[5] Li Chen, Chonghao Sima, Yang Li, Zehan Zheng, Jiajie Xu, Xiangwei Geng, Hongyang Li, Conghui He, Jianping Shi, Yu Qiao, et al. PersFormer: 3D Lane Detection via Perspective Transformer and the OpenLane Benchmark. *arXiv preprint arXiv:2203.11089*, 2022. 1

[6] Liang-Chieh Chen, Huiyu Wang, and Siyuan Qiao. Scaling wide residual networks for panoptic segmentation. *arXiv preprint arXiv:2011.11675*, 2020. 1

[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 5

[8] Olivier Faugeras and Quang-Tuan Luong. *The geometry of multiple images: the laws that govern the formation of multiple images of a scene and some of their applications*. MIT press, 2001. 2

[9] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018. 2

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 6

[11] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 2

[12] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020. 2

[13] Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. Single-View View Synthesis in the Wild with Learned Adaptive Multiplane Images. *arXiv preprint arXiv:2205.11733*, 2022. 2

[14] Adrian Johnston and Gustavo Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *CVPR*, 2020. 2

[15] Doyeon Kim, Woonghyun Ga, Pyungwhan Ahn, Donggyu Joo, Sehwan Chun, and Junmo Kim. Global-Local Path Networks for Monocular Depth Estimation with Vertical CutDepth. *arXiv preprint arXiv:2201.07436*, 2022. 2

[16] Varun Ravi Kumar, Sandesh Athni Hiremath, Markus Bach, Stefan Milz, Christian Witt, Clément Pinard, Senthil Yogamani, and Patrick Mäder. Fisheyedistancenet: Self-supervised scale-aware distance estimation using monocular fisheye camera for autonomous driving. In *ICRA*, 2020. 1, 2, 6

[17] Varun Ravi Kumar, Marvin Klingner, Senthil Yogamani, Markus Bach, Stefan Milz, Tim Fingscheidt, and Patrick Mäder. SVDistNet: Self-supervised near-field distance estimation on surround view fisheye cameras. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 2

[18] Varun Ravi Kumar, Stefan Milz, Christian Witt, Martin Simon, Karl Amende, Johannes Petzold, Senthil Yogamani, and Timo Pech. Monocular fisheye camera depth estimation using sparse lidar supervision. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018. 2, 6

[19] Varun Ravi Kumar, Senthil Yogamani, Markus Bach, Christian Witt, Stefan Milz, and Patrick Mäder. Unrectdepthnet: Self-supervised monocular depth estimation using a generic framework for handling common camera distortion models. In *IROS*, 2020. 2

[20] Varun Ravi Kumar, Senthil Yogamani, Hazem Rashed, Ganesh Sitsu, Christian Witt, Isabelle Leang, Stefan Milz, and Patrick Mäder. Omnidet: Surround view cameras based multi-task visual perception network for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):2830–2837, 2021. 1, 2, 3, 6, 7

[21] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 2, 6, 7

[22] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *ICCV*, 2021. 2

[23] Zhenyu Li, Zehui Chen, Xianming Liu, and Junjun Jiang. DepthFormer: Exploiting Long-Range Correlation and Local Information for Accurate Monocular Depth Estimation. *arXiv preprint arXiv:2203.14211*, 2022. 2

[24] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE TPAMI*, 2022. 2, 6

[25] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETRv2: A Unified Framework for 3D Perception from Multi-Camera Images. *arXiv preprint arXiv:2206.01256*, 2022. 1

[26] Yuxuan Liu, Yuan Yixuan, and Ming Liu. Ground-aware monocular 3d object detection for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):919–926, 2021. 3

[27] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Wei Li, Christian Theobalt, Ruigang Yang, and Wenping Wang. Adaptive surface normal constraint for depth estimation. In *ICCV*, 2021. 3

[28] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2018. 7

[29] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, 2018. 2, 3

[30] Vaishakh Patil, Christos Sakaridis, Alexander Liniger, and Luc Van Gool. P3Depth: Monocular Depth Estimation with a Piecewise Planarity Prior. In *CVPR*, 2022. 3

[31] Elad Plaut, Erez Ben Yaacov, and Bat El Shlomo. 3d object detection from a single fisheye image without a single fisheye training image. In *CVPR*, 2021. 1

[32] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 2

[33] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE TPAMI*, 2020. 2

[34] Ahmed Rida Sekkat, Yohan Dupuis, Varun Ravi Kumar, Hazem Rashed, Senthil Yogamani, Pascal Vasseur, and Paul Honeine. SynWoodScape: Synthetic Surround-View Fisheye Camera Dataset for Autonomous Driving. *IEEE Robotics and Automation Letters*, 7(3):8502–8509, jul 2022. 1, 2, 6

[35] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019. 7

[36] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *ICCV*, 1998. 2

[37] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 4

[38] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020. 1

[39] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 2

[40] Qingan Yan, Pan Ji, Nitin Bansal, Yuxin Ma, Yuan Tian, and Yi Xu. FisheyeDistill: Self-Supervised Monocular Depth Estimation with Ordinal Distillation for Fisheye Cameras. *arXiv preprint arXiv:2205.02930*, 2022. 2, 3

[41] Yaozu Ye, Kailun Yang, Kaite Xiang, Juan Wang, and Kaiwei Wang. Universal semantic segmentation for fisheye urban driving images. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 648–655, 2020. 1

[42] Senthil Yogamani, Ciarán Hughes, Jonathan Horgan, Ganesh Sistu, Padraig Varley, Derek O'Dea, Michal Uricár, Stefan Milz, Martin Simon, Karl Amende, et al. WoodScape: A multi-task, multi-camera fisheye dataset for autonomous driving. *arXiv preprint arXiv:1905.01489*, 2019. 5, 6

[43] Yifan Zhang, Qijian Zhang, Zhiyu Zhu, Junhui Hou, and Yixuan Yuan. GLENet: Boosting 3D Object Detectors with Generative Label Uncertainty Estimation. *arXiv preprint arXiv:2207.02466*, 2022. 1

[44] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *CVPR*, 2020. 1